RoGUE: RDMA over Generic Unconverged Ethernet

Yanfang Le

with Brent Stephens, Arjun Singhvi, Aditya Akella, Mike Swift
RDMA Overview

RDMA Overview Diagram:

- **RDMA**
  - USER
  - KERNEL
  - HARWARE

- **Zero Copy**
  - Application
    - Buffer
      - Kernel Bypass
  - Application
    - Buffer
      - Protocol Offload
RDMA Overview

Low Latency, High throughput, Low CPU utilization

Zero Copy

Kernel Bypass

Protocol Offload

RDMA

USER

KERNEL

HARWARE

Application

Buffer

Application

Buffer
RDMA Overview

- **RoCE**: a protocol that provides RDMA over a lossless Ethernet network.
RoCE assumes Ethernet network to be lossless - achieved by enabling Priority Flow Control (PFC).
RoCE assumes Ethernet network to be lossless - achieved by enabling Priority Flow Control (PFC).
Motivation
Motivation
Motivation
Motivation

HOL Blocking

Unfairness
Motivation

• Data center providers are reluctant to enable PFC
  – Instead, isolate RDMA traffic and TCP traffic
Motivation

• Data center providers are reluctant to enable PFC
  – Instead, isolate RDMA traffic and TCP traffic

• RDMA has not seen the uptake it deserves
Can we run RDMA over generic Ethernet network without any reliance on PFC?
Can we run RDMA over generic Ethernet network without any reliance on PFC?

RoCE + PFC
Congestion Control
No packet drop
Can we run RDMA over generic Ethernet network without any reliance on PFC?

RoCE + PFC
Congestion Control
No packet drop

RoGUE
Can we run RDMA over generic Ethernet network without any reliance on PFC?

**RoCE + PFC**
- Congestion Control
- No packet drop

**RoGUE**
- Congestion Control
Can we run RDMA over generic Ethernet network without any reliance on PFC?

RoCE + PFC
- Congestion Control
- No packet drop

RoGUE
- Congestion Control
- Retransmission
Can we run RDMA over generic Ethernet network without any reliance on PFC?

**RoCE + PFC**
- Congestion Control
- No packet drop

**RoGUE**
- Congestion Control
- Retransmission
- yet retain low latency, CPU utilization
RoCE Overview

- Verb
- RDMA APP
- Send QUEUE
- Receive QUEUE
- Completion QUEUE

CPU

RNIC

Brake the animations
RoCE Overview

RDMA APP

Send QUEUE

Receive QUEUE

Completion QUEUE

Verb

CPU

RNIC
RoCE Overview

RDMA APP

Send QUEUE

Receive QUEUE

Completion QUEUE

CPU

RNIC

Verb

Brake the animations
RoCE Overview

RDMA APP

Send QUEUE

Receive QUEUE

Completion QUEUE

Verb

CPU

Signal

RNIC

Brake the animations
Where to fix: HW or SW?

**Hardware**
- ✅ Low CPU utilization, Low Latency
- ❌ It requires to work with NIC vendor
- ❌ Heterogeneous network hardware with non-standard protocol implementation
- ❌ Complicates network evolution

**Software**
- ✅ Easy to implement
- ❌ Packet level congestion signals are unavailable
- ❌ High CPU utilization if per-packet operations
RoGUE Overview

- Congestion Control
- Loss Recovery

CPU

RNIC
RoGUE Overview

- Congestion Control
- Loss Recovery

- Congestion Control loop
- CPU-efficient segmenting

CPU
RNIC
RoGUE Overview

**Congestion Control**
- Congestion Control loop
- CPU-efficient segmenting

**Loss Recovery**
- Hardware timestamp to measure RTT
- Hardware rate limiter to pace packets

**CPU**

**RNIC**
RoGUE Overview

Congestion Control
- Congestion Control loop
- CPU-efficient segmenting

Loss Recovery
- Shadow Queue Pair

Hardware timestamp to measure RTT

Hardware rate limiter to pace packets
RoGUE Overview

Congestion Control
- Congestion Control loop
- CPU-efficient segmenting

Loss Recovery
- Shadow Queue Pair

CPU

Hardware timestamp to measure RTT

RNIC

Hardware retransmission

Hardware rate limiter to pace packets
Congestion Signal

Sender
Switch
Receiver

Packets from different flows
Congestion Signal

Packets from different flows
Congestion Signal

Sender

Switch

Receiver

RTT

ACK

Packets from different flows
Congestion Signal

Packets from different flows

Sender
Switch
Receiver

RTT

ACK

ACK

Packets from different flows
Congestion Signal

- RTT is high, the queue builds up, reduce the sending rate.
- RTT is low, network is idle, increase the sending rate.

Packets from different flows.
CPU Efficient Segmenting

• Two key questions
  • How large a verb should RoGUE send?
  • How often should the RNIC signaled?

• Small Verb (< 64KB)
  • signal every 64KB
  • CPU utilization (< 20%)

• Large Verb (>= 64KB)
  • chunk, and signal every 64KB.
  • CPU utilization (< 10%)
CPU Efficient Segmenting

- Two key questions
  - How large a verb should RoGUE send?
  - How often should the RNIC signaled?

- Small Verb (< 64KB)
  - signal every 64KB
  - CPU utilization (< 20%)

- Large Verb (>= 64KB)
  - chunk, and signal every 64KB.
  - CPU utilization (< 10%)
CPU Efficient Segmenting

- Two key questions
  - How large a verb should RoGUE send?
  - How often should the RNIC signaled?

- Small Verb (< 64KB)
  - signal every 64KB
  - CPU utilization (< 20%)

- Large Verb (>= 64KB)
  - chunk, and signal every 64KB.
  - CPU utilization (< 10%)
RTT measurement

- \( T_{enc\_s1} \)
- \( T_{enc\_s2} \)
- \( T_{comp\_s1} \)
- \( T_{comp\_s2} \)

Verb 1

Verb 1 packets

Signal 1

Send Ack 1

Send Ack 2
RTT measurement
RTT measurement

\[ T_{\text{start_si}} = \max( \text{Verb i enqueued, last packet of Verb i-1 goes out of NIC}) \]
RTT measurement

$T_{\text{enc}_s1}$

$T_{\text{enc}_s2}$

$T_{\text{start}_s2}$

$T_{\text{comp}_s1}$

$T_{\text{comp}_s2}$

$T_{\text{start}_s1}$ = max(
Verb i enqueued,
last packet of Verb i-1 goes out of NIC)
RTT measurement

\[
\begin{align*}
T_{\text{start}_s2} &= \text{max(Verb } i \text{ enqueued, last packet of Verb } i-1 \text{ goes out of NIC)} \\
RTT_i &= T_{\text{comp}_s_i} - T_{\text{start}_s_i} - \frac{\text{bytes}}{\text{rate_limit}}
\end{align*}
\]
RTT measurement

\[ T_{\text{start}_s2} = \max(\text{Verb } i \text{ enqueued, last packet of Verb } i-1 \text{ goes out of NIC}) \]

\[ \text{RTT}_i = T_{\text{comp}_s_i} - T_{\text{start}_s_i} - \frac{\text{bytes}}{\text{rate\_limit}} \]

RTT is measured by Hardware timestamp.
Congestion Response
Congestion Response

• Similar to TCP Vegas, and Timely
Congestion Response

• Similar to TCP Vegas, and Timely
• If congestion window >= 64KB, window-based + rate limiter
Congestion Response

- Similar to TCP Vegas, and Timely
- If congestion window $\geq 64$KB, window-based + rate limiter
- If congestion window $< 64$KB, rate limiter only
Congestion Response

• Similar to TCP Vegas, and Timely
• If congestion window $\geq 64$KB, window-based + rate limiter
• If congestion window $< 64$KB, rate limiter only
• Rate limiter is offloaded to RNIC
Evaluation

• Mellanox ConnectX-3 Pro 10Gbps RNICs, DCQCN
• Baselines: DCTCP, DCQCN
Evaluation-Cluster Experiments

- Each of 16 hosts generates 1MB RPC for random destinations and send 1KB RPC once every ten 1MB RPC

(a) Large RPCs (1MB) - Median FCT
(b) Small RPCs (1KB) - 90th %ile FCT
Evaluation-Congestion Response
Evaluation-CPU Utilization

![CPU Utilization Chart]

- DCTCP
- RoCE (READ RC)
- RoGUE (READ RC)
Summary

• It is possible to support RoCE without relying on PFC
• Judicious division of labor between SW and HW to do the congestion control and retransmission, yet retain a low CPU utilization
• RoGUE supports RC and UC transport types of CC
• Evaluation results validate that RoGUE has competitive performance with native RoCE